Contents

The **ServerConsole** is provided with the DevForce installation to host the *EntityServer* services in a console application.

You'll usually use the *ServerConsole* only during development and testing when your planned deployment is as a Windows Service. The *ServerConsole* is a simple utility you can stop and start as needed during development to test your n-tier application with a remote *EntityServer*.

The *ServerConsole* is not intended to be used in production deployments.

Files and assemblies

The following assemblies are always required:

IdeaBlade.Core.dll
IdeaBlade.EntityModel.dll
IdeaBlade.EntityModel.Edm.dll (version 7.2.2 and later: IdeaBlade.EntityModel.Edm.EF5.dll or
IdeaBlade.EntityModel.Edm.EF6.dll)
IdeaBlade.EntityModel.Server.dll
IdeaBlade.Linq.dll
IdeaBlade.Validation.dll

You'll also need the .config file:
ServerConsole.exe.config

If you're deploying a Code First model, you'll also need the following:

EntityFramework.dll
IdeaBlade.Aop.dll
PostSharp.dll

You'll of course also need all of your own "server-side" assemblies. These are the assemblies holding your entity model(s), and any custom extensions you've implemented.

Don't forget **.NET Framework 4.5**.

Configuration

Did you notice above that the configuration file used here is named ServerConsole.exe.config? This is because the executable is named ServerConsole.exe and we're following standard .NET config file naming and discovery conventions. (In Windows Server 2003, the config file should be named ServerConsole.config.)

The config file will contain most of what's in your app.config file in your client application. It's usually easiest to copy that file and edit as needed.

Database connection information

You'll usually need the <*connectionStrings*> for any databases your application uses. (You can also use <edmKeys> instead of or in addition to *connectionStrings*, but they may be deprecated in the future.)

If you are using a custom *DataSourceKeyResolver* to dynamically provide connection information, you will not need to define connection information in the config file.

Logging

You can use the <logging> element to set the file name and location of the debug log, or to turn logging off altogether. You also might want to archive log files. Generally other logging attributes are not needed, or used only for debugging purposes.

Logging on the server is usually a good idea, since the diagnostics provided will help during testing and in resolving deployment problems.

A typical config file might contain the following logging information:

```
<logging logFile="DebugLog.xml" archiveLogs="true" />
```

... to create a file named DebugLog.xml and archive files automatically.

Also see the Log topic for more information.

ObjectServer

You'll generally use the <objectServer> element to configure the service information for the *EntityServer*.

At a minimum you'll have something like the following:

```
<objectServer remoteBaseURL="http://localhost"
              serverPort="9009"
              serviceName="EntityService"
              >
</objectServer>
```

If you've modified the defaults for either *allowAnonymousLogin* or *loginManagerRequired* then you'll also need to include a <*serverSettings*> element with your settings. The <*clientSettings*> apply only to the client application.

Note above that we're using port 9009, one you often see in DevForce samples, but there are no DevForce requirements for this port number, and you can pick any free port. You should, however, not change the service name.

You will usually need to allow the *ServerConsole* to communicate through your firewall on the port you've chosen. For example, in Windows Firewall the first time you run ServerConsole.exe from a specific folder and for a particular port number you'll be prompted to unblock communications to allow the program to accept inbound communications on the port. You can also manually use your firewall software to open the port wanted, but this is less secure. See Windows Firewall documentation for more information.

Although we show use of the *http* protocol above, you can also use *net.tcp* and *https*. If using *https* you'll need to create an SSL certificate and map it to the port wanted - see this blog for more information.

As with any DevForce server or client, you can replace the <*objectServer*> configuration with a WCF <*serviceModel*> section which gives you complete control over the configuration of communications. See the samples.

Using the *ServerConsole*

You can find the *ServerConsole.exe* in the *Tools* subfolder under the DevForce installation. The executable does not have static references to DevForce assemblies, and can be used with any version of DevForce 2012.

All you need to do to use the *ServerConsole* is place all required files and assemblies in a folder, along with the ServerConsole.exe and ServerConsole.exe.config, and run the *ServerConsole* executable. As noted above you may get a firewall prompt, which you should accept, and the *EntityServer* services will start. The console window will display information about the service configuration, or error information if the service could not start.

You can then run your client application - on the same machine or another machine on the network. Remember to make sure that its <*objectServer*> settings match those of the server.

You can terminate the services by closing the window, or pressing either Enter or Ctrl-C within the window.

Note that by default the *ServerConsole* "publishes" its trace messages. See the logging topic for more information.

Troubleshooting

- You receive an *AddressAccessDeniedException* telling you that HTTP could not register your URL because you do not have access rights to the namespace. This is caused because the executable is not running with administrator privileges and HTTP addresses are secured resources. You have two options:
    1. Run the *ServerService* with an administrative account.
    2. Run the Netsh command line tool to register the namespace with the account. The steps are as follows:
        1. Open a command prompt using “Run as administrator” and enter:
        2. netsh http add urlacl url=http://+:9009/ user=DOMAIN\USERNAME
           ...where “9009” is the port you are using for the *EntityServer*, and DOMAIN\USERNAME is the system account to be granted access.
    3. Also see http://blogs.msdn.com/drnick/archive/2006/10/16/configuring-http-for-windows-vista.aspx for more information.